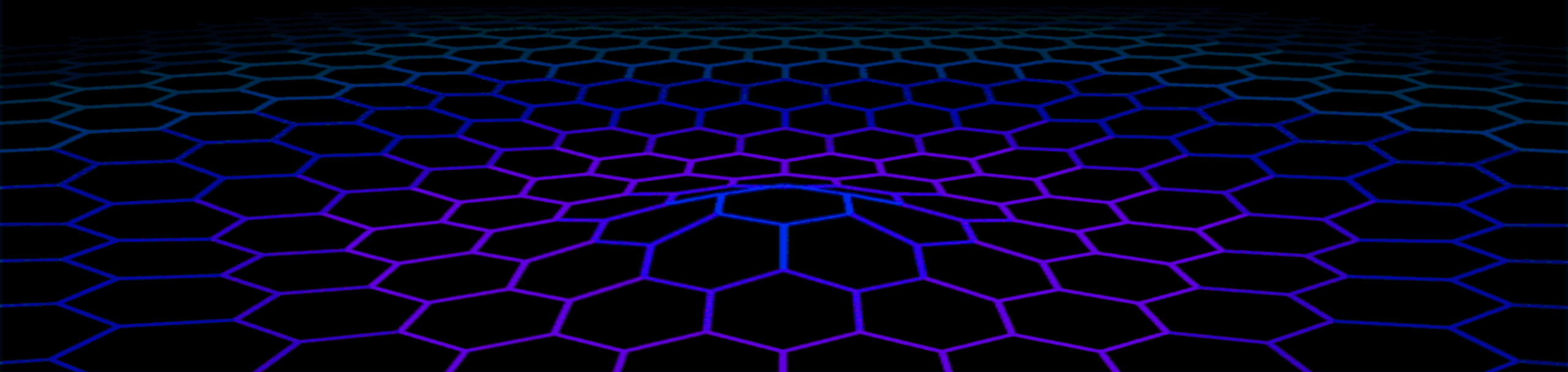
Unisenac
Campus Pelotas

# Banco de Dados 2



# CONTROLE DE ACESSO EM BANCO DE DADOS

# Controle de Acesso e Privilégios

No gerenciamento de bancos de dados, é essencial controlar quem tem acesso e quais operações cada usuário pode realizar.

Bancos de dados são amplamente utilizados em sistemas empresariais críticos, como ERP, CRM e e-commerce, onde é fundamental garantir a segurança e integridade dos dados.

MySQL e MariaDB oferecem comandos como GRANT, REVOKE e SHOW GRANTS para facilitar a gestão de privilégios.

Bora ver como criar usuários, conceder, visualizar e revogar privilégios, garantindo a segurança dos dados?

# Criação do Banco de Dados

```sql
-- MySQL
DROP DATABASE IF EXISTS aula09;
CREATE DATABASE aula09;
USE aula09;

-- PostgreSQL
DROP DATABASE aula09;
CREATE DATABASE aula09;
\c aula09
```

# Criação das Tabelas no MySQL

```sql
CREATE TABLE departamento (
  id   INT AUTO_INCREMENT PRIMARY KEY,
  nome VARCHAR(50) NOT NULL
) ENGINE=InnoDB;

CREATE TABLE funcionario (
  id             INT AUTO_INCREMENT PRIMARY KEY,
  nome           VARCHAR(100),
  email          VARCHAR(100),
  salario        DECIMAL(10,2),
  departamento_id INT,
  FOREIGN KEY (departamento_id) REFERENCES departamento(id)
) ENGINE=InnoDB;
```

# Criação das Tabelas no PostgreSQL

```sql
CREATE TABLE departamento (
  id   SERIAL PRIMARY KEY,
  nome VARCHAR(50) NOT NULL
);

CREATE TABLE funcionario (
  id             SERIAL PRIMARY KEY,
  nome           VARCHAR(100),
  email          VARCHAR(100),
  salario        NUMERIC(10,2),
  departamento_id INT,
  FOREIGN KEY (departamento_id) REFERENCES departamento(id)
);
```

# Inserção de Dados (MySQL e PostgreSQL)

```sql
INSERT INTO departamento (nome) VALUES
('TI'), ('RH'), ('Financeiro');

INSERT INTO funcionario (nome, email, salario, departamento_id) VALUES
('João Silva', 'joao@empresa.com', 4500.00, 1),
('Maria Souza', 'maria@empresa.com', 5200.00, 2),
('Carlos Lima', 'carlos@empresa.com', 6100.00, 1),
('Ana Paula', 'ana@empresa.com', 4000.00, 3);
```

# Criação de Usuários

```sql
-- MySQL
CREATE USER 'consultor'@'localhost' IDENTIFIED BY '1234';
CREATE USER 'gerente'@'localhost' IDENTIFIED BY '1234';

-- PostgreSQL
CREATE ROLE consultor WITH LOGIN PASSWORD '1234';
CREATE ROLE gerente WITH LOGIN PASSWORD '1234’;
```

Neste momento, o usuário ainda não tem permissão para acessar nenhum banco de dados.

Ao tentar fazer login, ele não conseguirá acessar o *shell* até que os privilégios sejam concedidos.

# Concedendo e Revogando Privilégios

GRANT: Atribui um privilégio a um usuário para realizar operações específicas.
REVOKE: Remove privilégios concedidos previamente a um usuário.

```
-- MySQL:
GRANT SELECT ON aula09.* TO 'consultor'@'localhost';
REVOKE SELECT ON aula09.* FROM 'consultor'@'localhost';
GRANT ALL PRIVILEGES ON aula09.* TO 'gerente'@'localhost';
REVOKE ALL PRIVILEGES ON aula09.* FROM 'gerente'@'localhost';

-- PostgreSQL:
GRANT SELECT ON ALL TABLES IN SCHEMA public TO consultor;
REVOKE SELECT ON ALL TABLES IN SCHEMA public FROM consultor;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO gerente;
REVOKE ALL PRIVILEGES ON ALL TABLES IN SCHEMA public FROM gerente;
```

# Criação de ROLES (Grupos de Permissões)

```sql
-- MySQL
CREATE ROLE analista;
GRANT SELECT, INSERT ON aula09.funcionario TO analista;
GRANT analista TO 'consultor'@'localhost';

-- PostgreSQL:
CREATE ROLE analista;
GRANT SELECT, INSERT ON funcionario TO analista;
GRANT analista TO consultor;

-- MySQL:
REVOKE analista FROM 'consultor'@'localhost';

-- PostgreSQL:
REVOKE analista FROM consultor;
```

# Após login como consultor

```
-- MySQL:
SET ROLE analista;

-- PostgreSQL:
SET ROLE analista;
```

# FLUSH – Recarregando privilégios

O comando FLUSH PRIVILEGES força o MySQL a recarregar as permissões em memória.

Quando usar?
Utilize este comando somente se as permissões forem modificadas manualmente nos arquivos de controle de usuários, como as tabelas mysql.user, mysql.db, etc.

Não é necessário usar FLUSH PRIVILEGES após os comandos GRANT ou REVOKE.
Esses comandos já atualizam automaticamente as permissões em tempo real.

Exemplo de uso:

```
FLUSH PRIVILEGES;
```

# Principais privilégios

CREATE - Criar novas tabelas ou bases de dados.
DROP - Excluir tabelas ou bases de dados.
SELECT - Realizar consultas SQL.
INSERT - Inserir registros em uma tabela.
UPDATE - Modificar dados existentes.
GRANT OPTION - Permite que um usuário conceda privilégios a outros usuários.

Usuários com GRANT OPTION:
- podem repassar apenas os privilégios que receberam.
- não podem criar usuários (a não ser que também tenham CREATE USER).
- não podem repassar privilégios que não têm.

Esses privilégios podem ser concedidos ou revogados com os comandos GRANT e REVOKE.

# Concedendo privilégios por campos

```
-- Em SGBDs como o MySQL, é possível restringir o acesso a colunas específicas ao conceder privilégios.

-- Sintaxe básica:
    GRANT <ação> (<colunas>) ON <banco>.<tabela> TO '<usuario>'@'<host>';

-- Exemplo 1: Acesso somente à coluna "mensagem"
    GRANT SELECT (mensagem) ON aula09.postagem TO 'pikachu'@'localhost';

-- Exemplo 2: Acesso às colunas "nome" e "email" da tabela "usuario"
    GRANT SELECT (nome, email) ON aula09.usuario TO 'pikachu'@'localhost';

-- Exemplo 3: Permitir UPDATE apenas nas colunas "email" e "fone"
    GRANT UPDATE (email, fone) ON aula09.usuario TO 'pikachu'@'localhost';

-- O PostgreSQL NÃO suporta esse nível de controle por coluna no GRANT padrão é necessário usar views ou Row-Level Security para alcançar o mesmo efeito.
```

# Testes de Permissão (como consultor)

```sql
-- Esperado: sucesso
SELECT * FROM funcionario;
INSERT INTO funcionario (nome, email, salario, departamento_id)
VALUES ('Novo Nome', 'teste@empresa.com', 3900.00, 2);

-- Esperado: erro
DELETE FROM funcionario WHERE id = 1;
```

# Visualização de Permissões

```sql
-- MySQL:
SHOW GRANTS FOR 'consultor'@'localhost';
SELECT CURRENT_ROLE();
SELECT CURRENT_USER();

-- PostgreSQL (dentro do psql):
\du consultor
SELECT current_user;
SELECT current_role;
```

# ALTERANDO NOME E SENHA DE USUÁRIO

```
-- Para alterar a senha de um usuário, utilize o comando ALTER USER.
-- Para alterar o nome de um usuário, utilize o comando RENAME USER.

Exemplos:

-- Alterando a senha do usuário 'joao':
ALTER USER 'joao'@'localhost' IDENTIFIED BY 'nova_senha123';

-- Alterando o nome do usuário 'joao' para 'novo_joao':
RENAME USER 'joao'@'localhost' TO 'novo_joao'@'localhost';

-- Alterando a senha e o nome do usuário ao mesmo tempo:
ALTER USER 'joao'@'localhost' IDENTIFIED BY 'senha123';
RENAME USER 'joao'@'localhost' TO 'joao_mudou'@'localhost';
```

# Remoção de Usuários e Roles

```sql
-- MySQL:
DROP USER 'consultor'@'localhost';
DROP USER 'gerente'@'localhost';
DROP ROLE analista;

-- PostgreSQL:
DROP ROLE consultor;
DROP ROLE gerente;
DROP ROLE analista;
```

# Fontes

```
-- Documentação oficial do MySQL:
https://dev.mysql.com/doc/refman/8.0/en/grant.html

-- Gerenciamento de usuários no MySQL:
https://dev.mysql.com/doc/refman/8.0/en/account-management.html

-- Documentação do MariaDB:
https://mariadb.com/kb/en/grant/

-- Documentação do PostgreSQL:
https://www.postgresql.org/docs/current/sql-grant.html

-- Gerenciamento de usuários no PostgreSQL:
https://www.postgresql.org/docs/current/user-manag.html
```

# -- Sabe onde tem exercícios (não avaliativos)?

-- Lá no **BlackBoard**